

Fig. 1

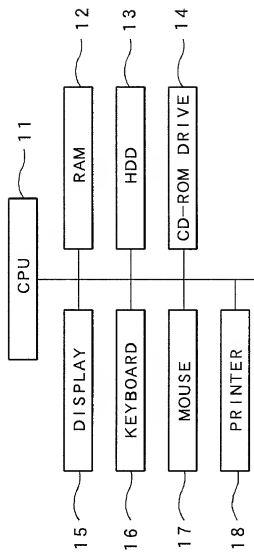
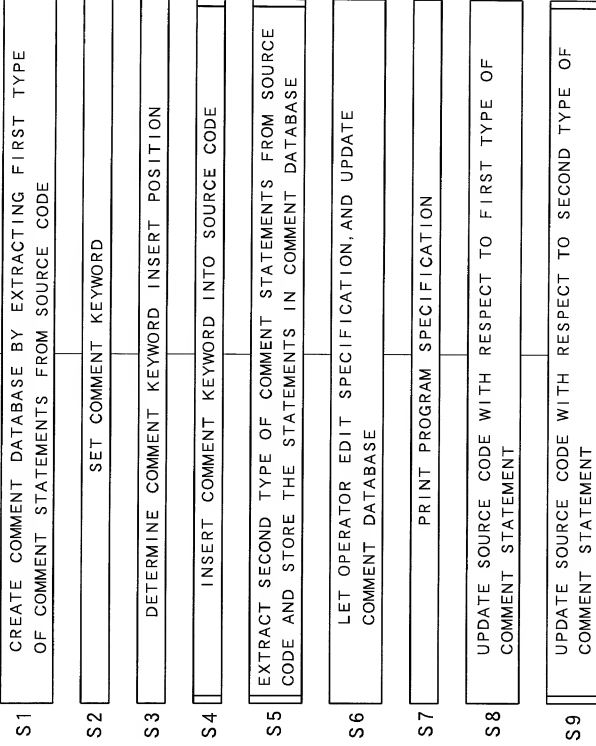


Fig. 2

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline()
{
    //initialize variable
    int c, nl;
    nl=0;
    // count number of lines till EOF is detected
    while ((c=getchar()) !=EOF)
        if (c=='\n')
            ++nl;
    // display line count on screen
    printf ("%d\n", nl);
    return nl;
}
```

Fig. 3

START



END

Fig. 4

IDENTIFIER	COMMENT ITEM	COMMENT
get line	OUTLINE	obtain line count of file
	RETURN VALUE	line count of file
	EXPLANATION OF PARAMETER	nothing
	FUNCTIONAL EXPLANATION	

Fig. 5

20

Comment Keyword Format [X]

☒ Extract Comment in Function (P)

Option

Definition of Comment Item

Function (E) : Functional Explanation

Member Function (M) : Functional Explanation

Definition of Keyword

Head Keyword (K) : *.*)

☐ Invalid (A)

Enclosure Keyword (S) :

Definition of Valid Column Position

Column Position (C) : 1 - 999

OK Cancel

21

Fig. 6

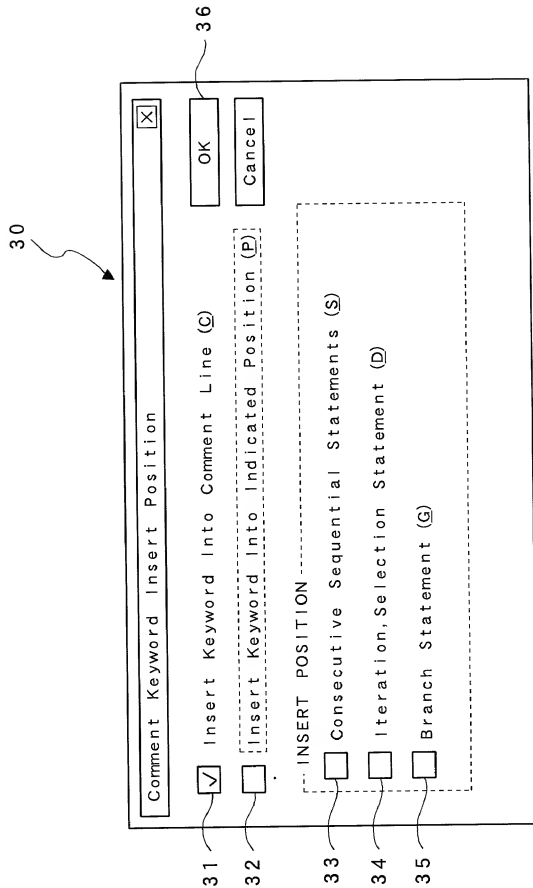


Fig. 7

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline()
{
    //initialize variable
    int c,nl;
    nl=0;
    //count number of lines till EOF is detected
    while((c=getchar())!=EOF)
        // check whether it is line feed signal or not
        if (c=='\n')
            // count number of lines in the case of line feed signal
            ++nl;
    // display line count on screen
    printf ("%d\n",nl);
    //return line count to caller
    return nl;
}
```

Fig. 8

30

Comment Keyword Insert Position

X

31 ☐ Insert Keyword Into Comment Line (C)

32 ☒ Insert Keyword Into Indicated Position (P)

33 ☒ Consecutive Sequential Statements (S)

34 ☒ Iteration, Selection Statement (D)

35 ☒ Branch Statement (G)

OK

Cancel

36

Fig. 9

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline()
{
    int c, nl;
    nl=0;
    while ((c=getchar()) !=EOF)
        if (c=='\n')
            ++nl;
    printf ("%d\n", nl);
    return nl;
}
```

Fig. 10

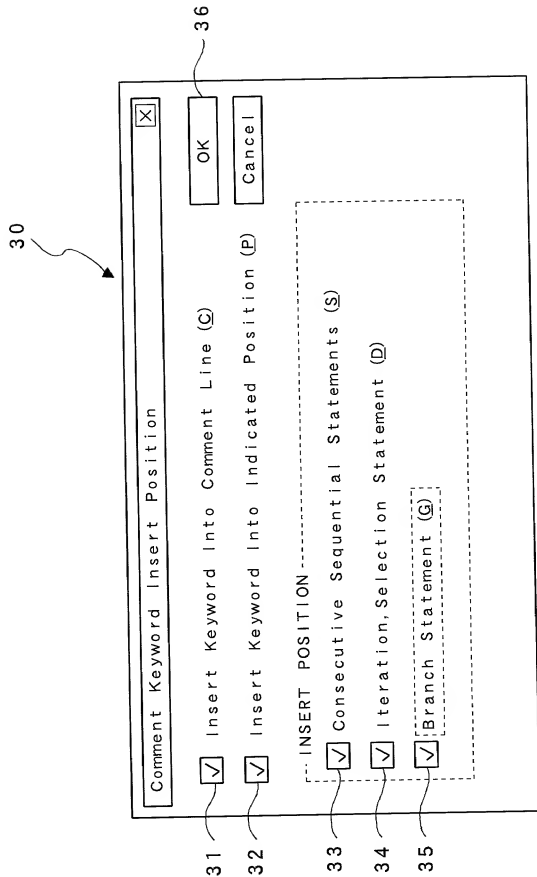


Fig. 11

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline()
{
    //1 initialize variable
    int c, nl;
    nl=0;
    //2 count number of lines till EOF is detected
    while ((c=getchar()) !=EOF)
        //2.1 check whether it is line feed signal or not
        if (c=='\n')
            //2.1.1 count number of lines in the case of line feed signal
            ++nl;
    //3 display line count on screen
    printf("%d\n", nl);
    //4 return line count to caller
    return nl;
}
```

Fig. 12

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline ()
{
    //1
    int c, nl;
    nl=0;
    //2
    while ((c=getchar ()) !=EOF)
        //2.1
        if (c=='\n')
            //2.1.1
            ++nl;
    //3
    printf ("%d\n", nl);
    //4
    return nl;
}
```

Fig. 13

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline ()
{
    //1 initialize variable
    int c,nl;
    nl=0;
    //2 count number of lines till EOF is detected
    while ((c=getchar ()) !=EOF)
        //2.1
        if (c=='\n')
            //2.1.1
            ++nl;
    //3 display line count on screen
    printf ("%d\n", nl);
    //4
    return nl;
}
```

Fig. 14

IDENTIFIER	COMMENT ITEM	COMMENT
	OUTLINE	obtain line count of file
	RETURN VALUE	line count of file
	EXPLANATION OF PARAMETER	nothing
	FUNCTIONAL EXPLANATION	1) initialize variable 2) count number of lines till EOF is detected 2. 1) 2. 1. 1) 3) display line count on screen 4)

40

Fig. 15

The diagram shows a window titled "Edit Screen" with a close button (X) in the top right corner. The window contains several sections:

- Explanation of getline Function**: A header section.
- Name of Function**: A field containing "getline".
- Definition File**: A field containing "Main.cpp".
- Definition Line Number**: A field containing "109".
- Declaration Format**: A field containing "int getline ()".
- Outline**: A section containing three items:
 - 41 obtain line count of file
 - 42 Return Value
 - 43 line count of file
- Explanation of Parameter**: A field containing "nothing".
- Functional Explanation**: A section containing a list of four items:
 - 1) initialize variable
 - 2) count number of lines till EOF is detected
 - 2. 1)
 - 2. 1. 1)
 - 3) display line count on screen
 - 4)

40

Fig. 16

Edit Screen
✕

Explanation of getline Function

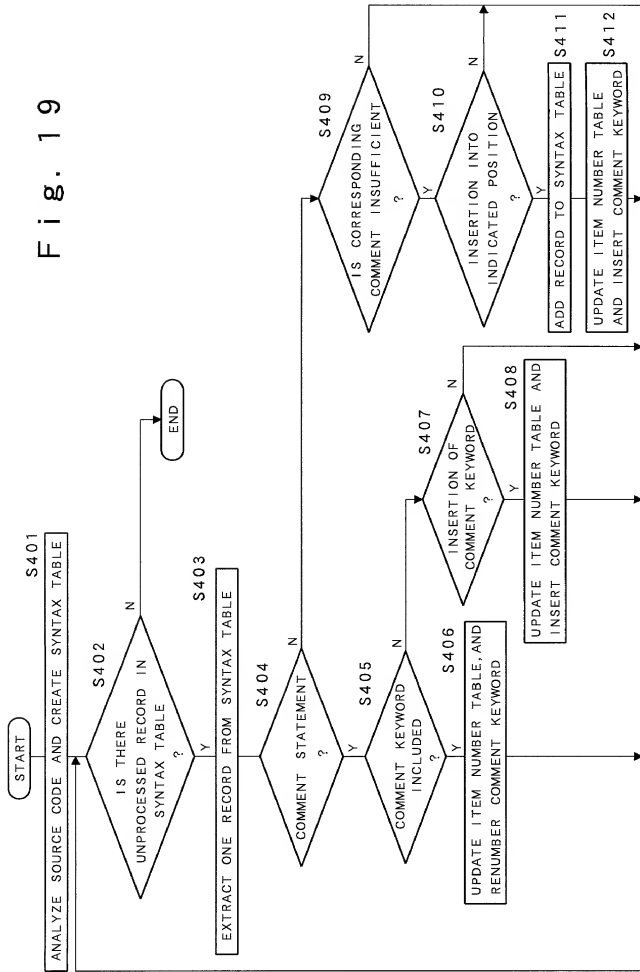
Name of Function	getline	
Definition File	Main.cpp	Definition Line Number 109
Declaration Format int getline ()		
Outline		
41 obtain line count of file		
42 Return Value line count of file		
Explanation of Parameter nothing		
Functional Explanation		
1) initialize variable 2) count number of lines till EOF is detected 2. 1) check whether it is line feed signal or not 2. 1. 1) count number of lines in the case of line feed signal 3) display line count on screen 4) return line count to caller		

Fig. 17

IDENTIFIER	COMMENT ITEM	COMMENT
getline	OUTLINE	obtain line count of file
	RETURN VALUE	line count of file
	EXPLANATION OF PARAMETER	nothing
	FUNCTIONAL EXPLANATION	1) initialize variable 2) count number of lines till EOF is detected
		2. 1) check whether it is line feed signal or not
		2. 1. 1) count number of lines in the case of line feed signal 3) display line count on screen 4) return line count to caller

```
//outline
// obtain line count of file
// return value
// line count of file
// explanation of parameter
// nothing
int getline ()
{
    //1) initialize variable
    int c, nl;
    nl=0;
    //2) count number of lines till EOF is detected
    while ( (c=getchar ()) !=EOF)
        //2.1) check whether it is line feed signal or not
        if (c=='\n')
            //2.1.1) count number of lines in the case of line feed signal
            ++nl;
    //3) display line count on screen
    printf (" %d\n", nl);
    //4) return line count to caller
    return nl;
}
```

Fig. 19



F i g . 20

STATEMENT CATEGORY	NEST NUMBER	STATEMENT
COMMENT STATEMENT	1	//) initialize variable
SEQUENTIAL STATEMENT	1	int c, nl;
SEQUENTIAL STATEMENT	1	nl=0;
COMMENT STATEMENT	1	//) count number of lines till EOF is detected
ITERATION STATEMENT	2	while (c=getchar()) !=EOF)
SELECTION STATEMENT	3	if (c=='\n')
SEQUENTIAL STATEMENT	3	++nl;
COMMENT STATEMENT	1	//) display line count on screen
SEQUENTIAL STATEMENT	1	printf ("%d\n", nl);
BRANCH STATEMENT	1	return nl;

Fig. 21

STATEMENT CATEGORY	NEST NUMBER	STATEMENT
COMMENT STATEMENT	1	//1) initialize variable
SEQUENTIAL STATEMENT	1	int c, nl;
SEQUENTIAL STATEMENT	1	nl=0;
COMMENT STATEMENT	1	//2) count number of lines till EOF is detected
ITERATION STATEMENT	2	while ((c=getchar()) !=EOF)
COMMENT STATEMENT	2	//2.1)
SELECTION STATEMENT	3	if (c=='\n')
COMMENT STATEMENT	3	//2.1. 1)
SEQUENTIAL STATEMENT	3	++nl;
COMMENT STATEMENT	1	//3) display line count on screen
SEQUENTIAL STATEMENT	1	printf ("%d\n", nl);
COMMENT STATEMENT	1	//4)
BRANCH STATEMENT	1	return nl;

Fig. 22

```
//outline
//  obtain line count of file
//  return value
//  line count of file
//  explanation of parameter
//  nothing
int getline ()
{
    //2) initialize variavle
    int c, nl;
    nl=0;
    //3) count number of lines till EOF is detected
    while ((c=getchar ()) !=EOF)
        if (c=='\n')
            ++nl;
    //6) display line count on screen
    printf ("%d\n", nl);
    return nl;
}
```

Fig. 23

STATEMENT CATEGORY	NEST NUMBER	STATEMENT
COMMENT STATEMENT	1	//2) initialize variable
SEQUENTIAL STATEMENT	1	int c, nl;
SEQUENTIAL STATEMENT	1	nl=0;
COMMENT STATEMENT	1	//3) count number of lines till EOF is detected
ITERATION STATEMENT	2	while ((c=getchar()) != EOF)
SELECTION STATEMENT	3	if (c == '\n')
SEQUENTIAL STATEMENT	3	++nl;
COMMENT STATEMENT	1	//6) display line count on screen
SEQUENTIAL STATEMENT	1	printf ("%d\n", nl);
BRANCH STATEMENT	1	return nl;

Fig. 24

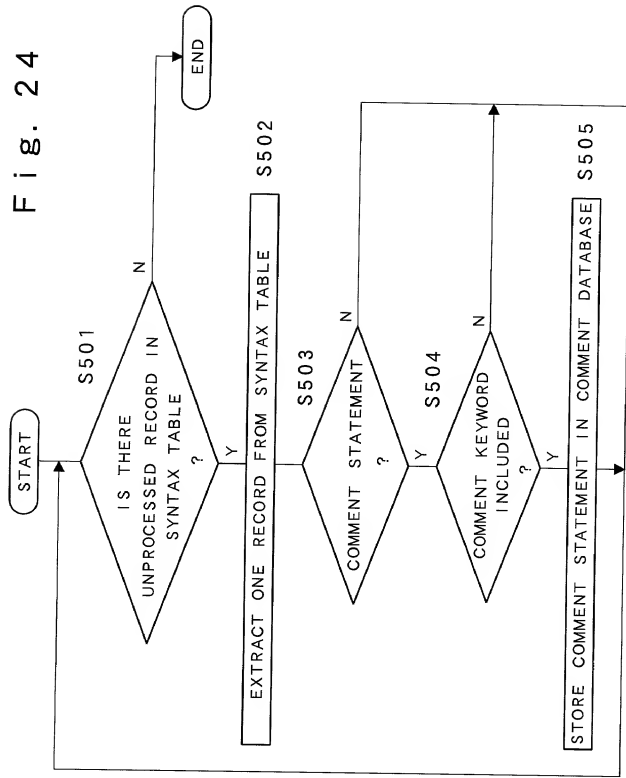


Fig. 25

